Servicios Web

Módulo 2

Bruno Mendoza Guedes
http://www.brunomendoza.es
hola@brunomendoza.es
@_brunomendoza

Definición y características

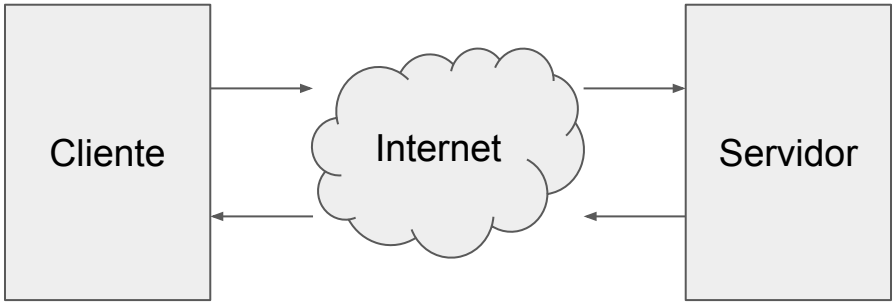
Introducción

- En los comienzos de la Web las páginas eran sencillas. Consistían en texto y enlaces.
- Con el paso del tiempo los documentos HTML fueron dotados con la capacidad de traer recursos en segundo plano.
- El navegador introdujo el sistema Asynchronous JavaScript and XML o AJAX para traer datos del servidor sin tener que refrescar la página Web completamente.

Introducción: Servicio Web

Un **servicio Web** es un sistema de comunicación entre dos máquinas (cliente-servidor) que permite la solicitud y entrega de mensajes.

- Esta comunicación se produce a través de Internet.
- El protocolo HTTP se utiliza para traer un documento HTML o otro tipo de datos.



Introducción: API de un Servicio Web

Los servicios Web se consumen a través de una interfaz (Application Program Interface o API) que puede ser diferente entre dos o más servicios web.

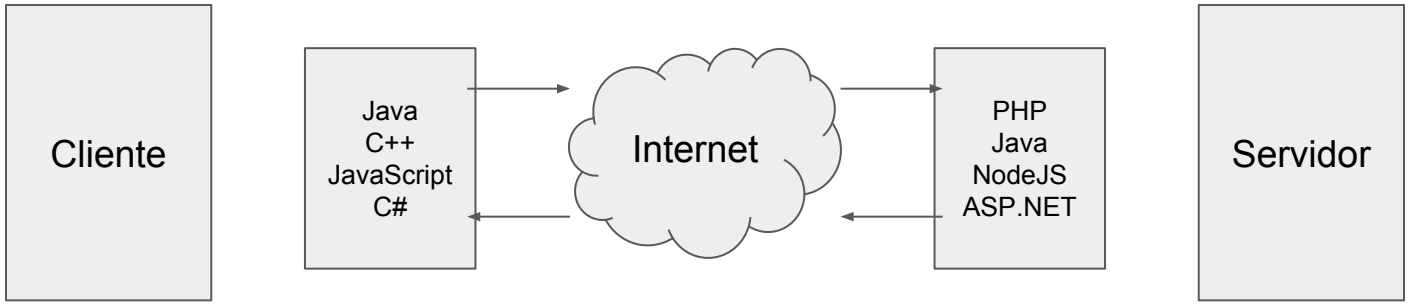
La API define las herramientas o el lenguaje necesario para consumir el servicio Web.

Introducción: Elementos de la API

- La sintaxis puede estar compuesta por métodos como los utilizados en cualquier lenguaje de programación, los métodos definidos en la especificación HTTP o que hago uso de URIs.
- Los nombres y valores de los tipos de datos.
- La autenticación si la hubiera.
- El formato de los mensajes devueltos por el servidor. Por ejemplo, POX, SOAP o JSON.
- Parámetros o tipos de datos en forma de clave-valor (metadata).

Introducción: Encapsulamiento

El acceso al servicio Web se puede alcanzar a través de una librería o a través de la propia especificación de la API.



Introducción: Conclusión

- El acceso al servicio Web se puede alcanzar a través de una librería o a través de la propia especificación.
- La API y la documentación son críticos para que este servicio pueda ser utilizado.
- Esto permite que los servicios Web puedan ser utilizados por diferentes dispositivos y tecnologías. Por ejemplo, un servicio Web podría ser consumido por un teléfono móvil que utilizara Android y por un frigorífico que utilizara Java ME sin modificar los mensajes.

Historia: Predecesores

- El término servicio Web en el contexto que nos encontramos es anterior a la Web.
- Uno de los primeros servicios que utiliza la arquitectura de un servicio Web es UDDI, acrónimo de Electronic Data Interchange.
- EDI fue definido en 1996 por el National Institute of Standards y Tecnologías pero su existencia e implementación data de los 60.

https://en.wikipedia.org/wiki/Electronic_data_interchange

Historia: RPC

- En los 90 surgió RPC o Remote Procedure Control.
- RPC está basado en DCE o Distributed Computing Environment.
- RPC permite llamar métodos o funciones disponibles en computadoras remotas.
- Tiene sus raíces en UNIX. Microsoft creo su propia versión que llamó MSRPC a la que siguió la arquitectura Microsoft COM (Common Object Model) OLE (Object Linking and Embedded) que está relacionado con DCOM (Distributed Component Object Model).
- NIS o NFS utilizan esta tecnología

https://en.wikipedia.org/wiki/Remote_procedure_call

Historia: CORBA

- También en los noventa surgió la especificación CORBA o Common Object Request Broker Architecture.
- Su ventaja principal consiste en la interoperabilidad. Permite que un conjunto de máquinas puedan comunicarse con independencia del sistema donde este estuviera implementado.

https://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture

Historia: RMI

RMI o Remote Method Invocation es un simplificación de CORBA que utiliza el protocolo Inter ORB o IIOP.

Entre 1991 y 1996 se diseñaron multitud de protocolos de comunicación entre computadoras.

https://en.wikipedia.org/wiki/Java_remote_method_invocation

Historia: XML

- En 1998 apareció XML o Extensible Markup Language, que deriva de la especificación SGML o Standard Generalized Markup Language.
- Los objetivos que persigue XML son la simplicidad, legibilidad y la generalización.
- XML tiene formato de texto y no binario.
- XHTML es una derivación de XML.
- XML es ideal como medio de transporte durante el intercambio de cualquier información.

Historia: XML-RPC

- Emergieron entonces otros formatos de archivo como XML-RPC que consiste en un conjunto de reglas que definen un conjunto de tipos de datos que pueden ser transmitidos a través de la Web.
- ATOM y RSS son especificaciones muy populares de XML utilizadas en servicios Web actuales.
- Poco después de la aparición de XML apareció SOAP para revolucionar el uso de los servicios Web.
- SOAP o Simple Object Abstract Notation también es una especificación basada en XML y el estándar XML-RPC.
- Esta especificación se convirtió en la base de un conjunto de tecnologías Web que se llamaron WS-* o Web Services Everything.

https://en.wikipedia.org/wiki/XML-RPC

Historia: JSON

- Y el formato de mensajes más reciente y avanzado es JSON o Javascript Simple Object Notation.
- JSON apareció a principios los años que siguieron al 2000 pero realmente ha ganado popularidad en los últimos años.
- Como XML, JSON es un formato de texto y legible para humanos, pero su notación es mucho más simple.
- JSON ha desplazado a SOAP en entornos de baja escala. SOAP sigue dominando en entornos empresariales.

Formatos: Herramientas

- Para trabajar con los mensajes usaremos un herramientas software para no tratar los mensajes directamente y un depurador.
- **Postman** es una herramienta gratuita y completa.
- Otra opción es hacerlo a través de **Curl**. Bastante más complejo pero completo.

Formatos

Los formatos más comunes son:

- ATOM
- RSS 2.0
- POX o Plain Old XML
- JSON
- SOAP

Formatos: SOAP

Formato de petición basado en XML:

```
<?xml version="1.0">
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Body xmlns:m="http://www.example.org/stock">
      <m:GetStockPrice>
        <m:StockName>IBM</m:StockName>
      </m:GetStockPrice>
    </soap:Body>
</soap:Envelope>
```

Formatos: SOAP

Es necesario utilizar el método POST en la petición HTTP.
Dada la complejidad de SOAP se utilizan librerías el lado del servidor y el lado del cliente.

Formatos: SOAP

Respuesta SOAP:

```
<?xml version="1.0">
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Body xmlns:m="http://www.example.org/stock">
      <m:GetStockPriceResponse>
        <m:Price></m:Price>
      </m:GetStockPriceResponse>
    </soap:Body>
</soap:Envelope>
```

Formatos: Otros

- AMF o Action Message Format. Es un formato binario diseñado para Flash.
- CSV o Comma Separated Values.
- Binarios
- Hojas de estilo
- Bases de datos empaquetadas
- Imágenes

Tipos de Servicios Web

Estándares

Existen muchos estándares. Los más populares son:

- REST
- SOAP
- OData

Estándares: SOAP

- Conjunto específico de elementos, atributos y tipos de datos.
- Con SOAP se introdujo SOA o Service Oriented Architecture.
- SOAP es la base de los estándares WS-*
- Fue impulsado por Microsoft y gestionado por W3C desde 2001.
- Hay que distinguir el formato de mensaje SOAP y software subyacente.
    - Remote Object Proxy Engine o ROPE. Componentes COM (soap.dll) que asisten durante la construcción de mensajes SOAP en nuestra aplicación.

Estándares: SOAP

- No se necesita trabajar directamente con los archivos XML.
- Se utilizan 'operaciones' que son definidas por el servicio utilizado.
- SOAP presenta algunos inconvenientes como:
  - su verbosidad,
  - el uso de librerías necesarias para crear las peticiones y las respuestas
  - y convertir los tipos de datos a XML y viceversa.

Estándares: REST

- Hace unos años muchos de los servicios Web SOAP pasaron a ser servicios Web RESTful.
- REST o Representational State Transfer fue diseñado sobre la misma época que SOAP por Roy Fielding que también es uno de los autores principales de HTTP.
- REST es una arquitectura no un formato de mensaje. Se fundamenta en peticiones y respuestas HTTP simples.
- Los servicios RESTful utilizan los métodos GET, POST, PUT y DELETE.
- Los servicios RESTful pueden no utilizar los mismos formatos de mensajes.

Estándares: REST

Restricciones:

- Los servicios RESTful son por definición 'stateless' (sin estado). Todas las peticiones son independientes.
- Las ventajas de SOAP como, por ejemplo, los tipos de datos, la interoperabilidad o la seguridad son las razones por las que se siguen utilizando en entornos computacionales de gran escala.

Estándares: OData

- OData está basado en REST: solicita los datos a través de URI y obtiene los datos en formatos traducibles (parsing).
- Como en SOAP, OData tiene un estricto conjunto de reglas relacionadas con los nombres de los elementos y reglas sobre el formato.
- Como en SOAP, OData ofrece librerías para gestionar los mensajes.
- Los mensajes de OData pueden estar definidos en ATOM o JSON.
- OData comenzó como una tecnología propietaria de Microsoft, ahora es Open Source y tiene soporte en multitud de lenguajes y plataformas.

Estándares: Otros

- Otros estándares de servicios Web son: XML-RPC, UDDI o Universal Description Discovery and Integration, WSDL o Web Service Description Language.
- Este último se utiliza para describir servicios basados en SOAP. UDDI se utiliza para encontrarlos y WSDL para hacerlos utilizables.
- WSDM o Web Services Distributed Management es un estándar para gestionar y monitorizar el estado de otros servicios.
- WS-* o Web Services Everything. Son cientos de estándares que se pueden encontrar en http://www.ws-i.org
- Los estándares de servicios Web describen lenguajes basados en XML para gestionar procesos de negocio y transacciones, seguridad, confianza y gestión.

Directorios

- Registro de servicios Web. Un registro de servicios Web podría ser un directorio donde cada servicio describiría sus capacidades a través de otro lenguaje basado en XML.
- La especificación de este lenguaje está definida en el estándar UDDI (Universal Description Discovery and Integration).
- UDDI fue desarrollado en el 2000 como parte del estándar WS-I o Web Service Interoperability.
- UDDI ya no se utiliza en la Web pública.

Directorios

Enlaces útiles:

- http://www.programmableweb.com
- http://www.webservicex.net/ws/default.aspx

SOAP

Introducción

Introducción

- Acrónimo de **Simple Object Access Protocol** aunque hoy tiene un significado diferente.
- SOAP es un lenguaje basado en XML que representa mensajes de petición y respuesta entre cliente y servidor respectivamente.
- SOAP fue desarrollado por Microsoft a finales de los 90 por un grupo de desarrolladores liderados por Dave Winer.

Introducción

- Desde el 2000 ha sido gestionado por el W3C. La estandarización llegó con la versión 1.2 en forma de recomendación por la W3C en 2003.
- Es la base de un conjunto completo de especificaciones llamada WS-*

https://msdn.microsoft.com/en-us/library/ms951274.aspx

Objetivos

SOAP persigue tres objetivos fundamentales:

- Extensibilidad,
- neutralidad e
- independencia.

Objetivos

SOAP persigue tres objetivos fundamentales:

- Extensibilidad,
- neutralidad e
- independencia.

Objetivos

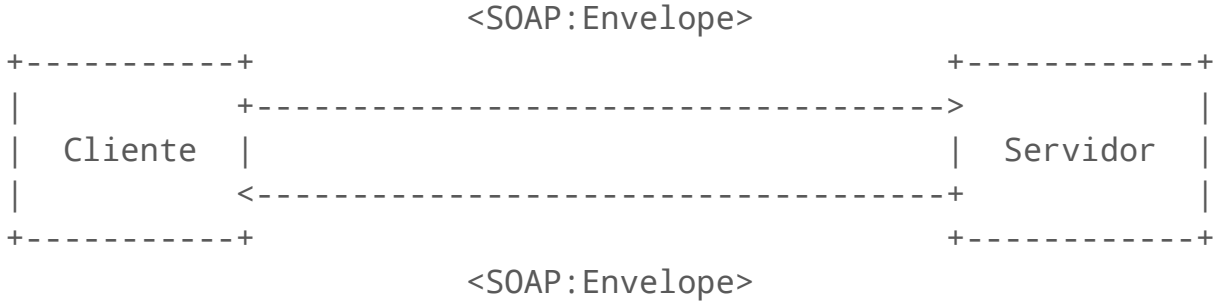
- La extensibilidad se refiere a la capacidad de crecimiento.
- La neutralidad se refiere a la capacidad de poder ser utilizado sobre cualquier protocolo de transporte.
- Y la independencia se refiere a la capacidad de ser utilizado bajo cualquier sistema o lenguaje programación.

Versiones

- Existen dos versiones de SOAP: SOAP 1.1 y SOAP 1.2.
- La segunda versión se convirtió en un estándar aunque, para incrementar la interoperabilidad entre cliente y servidor, muchas plataformas de desarrollo soportan ambas versiones.
- SOAP 1.2 tiene muchas ventajas, incluyendo un procesamiento más claro, modelos de extensibilidad y una mejor integración Web.

Sistema

- Cuando un programador utiliza SOAP no necesita implementar métodos o algoritmos de serialización (marshalling o proceso de transformación de representaciones en memoria de un objeto a un formato de datos adecuado para el almacenamiento o transmisión) de los mensajes XML.
- Es necesario el uso de librerías compatibles con la plataforma y lenguaje actual tanto en el lado del servidor como en el lado del cliente para realizar las traducciones.



Sistema

- Debes indicar que servicio quieres utilizar procurando un documento WDSL.
- Entonces llamarías a los objetos y métodos de servicios remotos utilizando las librerías o clases que has implementado.
- Los mensajes se envían y reciben dentro de un 'sobre' (envelope).
- El sobre de petición contiene las órdenes a ejecutar y el sobre de respuesta contiene los datos retornados.
- Si se produce un problema la respuesta será un mensaje de falta (error).
- Los datos enviados por el servidor son traducidos por las librerías utilizadas en el cliente.

Ventajas

- Soportada por muchas plataformas y lenguajes.
- Libera a los programadores de tener que enfrentarse directamente con la creación y lectura de los mensajes del servicio Web.
- Funciona con infraestructura utilizada en Internet: HTTP.
- Su desventaja principal es el tamaño de los mensajes: tiempo y ancho de banda. Además de la necesidad de utilizar librerías tanto en el lado del servidor como en el lado del cliente.

Peticiones y respuestas

Formato

- Un mensaje de servicio Web basado en SOAP es un paquete jerárquico.
- Lo componen hasta en cuatro partes.
- Las dos primeras son obligatorias.
- El elemento raíz se llama 'Envelope'.
- La etiqueta no es importante. Sí lo es el espacio de nombres utilizado.

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope">

    ...

</env:Envelope>
```

Formato

El elemento **Envelope** puede contener dos descendientes: **Header** y **Body**.
Header es opcional y *Body* es obligatorio.
Header contendrá metadatos, por ejemplo llaves (key) de la API, credenciales de registro para validar las peticiones o timestamps.

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <n:alertcontrol xmlns:n="http://example.com/stock">
            <n:priority>1</n:priority>
            <n:expires>2017-12-04T15:15-00:00</n:expires>
        </n:alertcontrol>
    </env:Header>

    ...
```

Formato

Ejemplo:

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <n:alertcontrol xmlns:n="http://example.com/stock">
            <n:priority>1</n:priority>
            <n:expires>2017-12-04T15:15-00:00</n:expires>
        </n:alertcontrol>
    </env:Header>
    <env:Body>
        <m:GetStockPrice xmlns:m="http://www.example.org/stock">
            <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
    </env:Body>
</env:Envelope>
```

Formato

- *GetStockPrice* es el procedimiento remoto definido por el servicio. Y los elementos descendientes representan parámetros para este procedimiento.
- Los nombres de los procedimientos y sus parámetros se definen en el documento WSDL.
- El mensaje de respuesta también tiene un elemento raíz *Envelope*. Como en la petición, la respuesta necesita un espacio de nombres para indicar la versión.
- La respuesta también necesita un elemento *Body*.
- El espacio de nombres del cuerpo es único para el servicio Web que estemos utilizando.

Formato

La respuesta, como la petición, también puede contener sólo el elemento *Body*:

```
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Body>
        <m:GetStockPriceResponse xmlns:m="http://www.example.org/stock">
            <m:Price>99.9</m:Price>
        </m:GetStockPriceReponse>
    </env:Body>
</env:Envelope>
```

Formato

- El tipo del dato se define en el documento WSDL.
- Si la petición falla se envía una respuesta conteniendo el elemento **Fault** como elemento descendiente del elemento *Body*.
- En SOAP 1.2 el elemento *Fault* contiene los siguientes elementos
  - Code,
  - Reason,
  - Role y
  - Detail.

Formato

Ejemplo de error:

```
<env:Body>
  <env:Fault>
    <env:Code><env:Value>env:Sender</env:Value></env:Code>
    <env:Reason><env:Text xml:lang="en-US">Message</env:Text></env:Reason>
    <env:Role>http://example.com/stock</env:Role>
    <env:Detail></env:Detail>
  </env:Fault>
</env:Body>
```

Tipos de datos

Estandarización

- Para asegurar la compatibilidad es necesario que las librerías sigan la recomendación publicada por la W3C.
- Una de las recomendaciones más importantes es el tipo de datos que codificar en un mensaje SOAP.

https://www.w3.org/TR/xmlschema-2/

Tipos de datos

- SOAP soporta un amplio número de tipos de datos.
- El desarrollador no necesita que utiliza las librerías no necesita trabajar con estos tipos aunque si quien implemente una librería que funcione como traductor (parser).
- Existen lenguajes como, por ejemplo, Java que tienen casi una correspondencia total con los tipos de datos que ofrece SOAP. El resto tendrá que realizar una traducción.

Tipos de datos

Los tipos de datos utilizados se especifican en el documento WSDL.

```
<element name="id" type="xsd:integer">
<element name="name" type="xsd:string">
<element name="age" type="xsd:integer">
```

https://www.w3.org/TR/xmlschema-2/#built-in-datatypes

Tipos de datos

SOAP también permite crear objetos complejos.

A veces llamados *Value Object* o *Data Transfer Object*.

```
<element name="Book">
   <complexType>
      <element name="author" type="xsd:string"/>
      <element name="preface" type="xsd:string"/>
      <element name="intro" type="xsd:string"/>
   </complexType>
</element>
```

Tipos de datos

El elemento *complexType* es parte del dialecto WSDL.

Uso en un mensaje SOAP:

```
<e:Book>
    <e:author>John Doe</e:author>
    <e:preface>Some preface goes here</e:preface>
    <e:intro>This is a very little introduction, isn't it?</e:intro>
</e:Book>
```

El espacio de nombres e apunta al documento WSDL definido en el servicio.

Tipos de datos

También es posible utilizar arrays (estructura homogénea).

```
<complexType name="ArrayOfFloat">
    <complexContent>
        <restriction base="soapenc:Array">
            <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:float[]"/>
        </restriction>
    </complexContent>
</complexType>
```

La restricción dice que es un array y el atributo dice que es de *floats*.

Tipos de datos

El elemento *parte de mensaje* en el documento WSDL puede declarar el uso de este tipo.

```
<message name="GetTradePriceOutput">
    <part name="result" type="ArrayOfFloat"/>
    <part name="frequency" type="xsd:float"/>
</message>
```

El nombre de este mensaje es *GetTradePriceOutput* y tiene una parte que utiliza el tipo de datos *ArrayOfFloat*.

Tipos de datos

- *message* describe los datos que se van a intercambiar entre el proveedor y el consumidor del servicio Web.
- Cada servicio Web tiene dos mensajes. Uno de salida y otro de entrada.
- La entrada describe los parámetros para el servicio Web y la salida describe los datos devueltos por el servicio Web.
- Cada mensaje contiene cero o más elementos *part*, uno por cada parámetro de la función, procedimiento o método del servicio Web.
- Cada elemento *part* está asociado con un tipo de datos concreto definido en el elemento *types*.

Tipos de datos

Cuando los datos se codifican en un mensaje SOAP, tendrá el nombre que se especifica en el documento WSDL.

```
<m:GetTradePriceOutput>
    <array soapenc:arrayType="xsd:float[]">
        <item>34.5</item>
        <item>34.6</item>
        <item>34.7</item>
    </array>
</m:GetTradePriceOutput>
```

Tipos de datos

- SOAP puede representar casi cualquier cosa.
- Pero recuerda, no tienes que codificar los mensajes SOAP.

Documento WDSL

WSDL

- WSDL o Web Service Description Language.
- Es un dialecto XML.
- Es un documento que describe la funcionalidad y uso del servicio Web.
- Diseñado para que sea leído por software.
- Al ser XML también es legible por humanos.
- Muchos de estos documentos son generados automáticamente por el servidor.

https://www.w3.org/TR/wsdl20/

WDSL

El elemento raíz de un documento WSDL es *definitions*.

```
<definitions name="HelloService"
    targetSpace="http://www.examples.com/wdsl/HelloService.wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://examples.com/wsl/HelloService.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>

    ...

</definitions>
```

WDSL

Espacios de nombre que debemos definir:

- targetNamespace es el espacio de nombres asociado con el servicio Web.
- El espacio de nombres de WDSL para los elementos incluidos en el documento.
- El espacio de nombres para elementos de SOAP.
- El espacio de nombres del servicio Web nuevamente.
- Y el espacio de nombres para elementos del esquema XML.

WDSL

Ejemplos:

https://www.w3.org/2001/04/wsws-proceedings/uche/wsdl.html

http://www.webservicex.net/WeatherForecast.asmx?WSDL

Cuando no no se incluye la etiqueta en en la definición de un espacio de nombres significa que es el espacio de nombres por defecto.

WDSL

- El elemento *documentation* describe el servicio que ofrece. Es opcional.
- El elemento *types* es obligatorio y define los tipos de datos que hemos implementado.
- El elemento *sequence* dentro del elemento *complexType* define un conjunto de propiedades.
- El siguiente elemento dentro del elemento *definitions* es *message*. Pueden existir uno o más elementos de tipo *message*. Estos definen las funciones disponibles en el servicio Web.

WDSL

- El elemento *portType* sirve para declarar operaciones. Puede contener lo elementos *documentation*, *input* y *output*. Estos dos últimos elementos representan mensajes.
- Luego debemos declarar el elemento *binding*. Este elemento puede indicar la versión de SOAP al cliente o las operaciones disponibles.
- El último elemento es *service*. Puede incluir el elemento *documentation* y elementos *port* que fueron definidos en el elemento *binding*.

WDSL

- Los IDEs, por ejemplo Visual Studio o Eclipse, pueden leer el documento WSDL y generar un conjunto de clases proxy que utilizará el cliente del servicio Web.
- Todo el trabajo de serialización y deserialización será cosa de estas clases.

OData

OData

- OData o Open Data Protocol
- Open Data sigue la arquitectura REST.
- Espera una petición formada como URI en vez de XML o JSON y, en sus versiones más longevas, puede devolver datos en los formatos XML o JSON.
- Pero OData trabaja con ATOM.
- Al utilizar un tipo de formato específico es posible crear librerías o conjuntos de herramientas para el desarrollo de aplicaciones que evitan tener que traducir los mensajes.

OData

- OData fue ideado por Microsoft.
- Continuan integrando esta tecnología en sus servicios basados en la Nube como, por ejemplo, Azure, SQL Azure (DBaaS o Sharepoint.
- Es un estándar Open Source.
- Hay implementaciones para varios sistemas operativos y lenguajes de programación.
- La última versión es OData 4.0

http://www.odata.org/getting-started/understand-odata-in-6-steps/

Componentes

- El modelo de datos es un estándar de formato de mensajes. Sirve para organizar y describir los datos.
- OData utiliza el concepto EDM o Entity Data Model.

http://www.odata.org/documentation/odata-version-3-0/common-schema-definition-language-csdl/

Componentes

- El modelo de datos es un estándar de formato de mensajes. Sirve para organizar y describir los datos.
- OData utiliza el concepto EDM o Entity Data Model.
- Sirve para representar datos estructurados a través de un vocabulario consistente.
- Este lenguaje puede ser codificado en los formatos ATOM o JSON.

http://www.odata.org/documentation/odata-version-3-0/common-schema-definition-language-csdl/

Componentes

- Las peticiones se realizan generalmente a través de HTTP pero, a diferencia de los servicios RESTful, OData sigue normas para construir la URI.
- También utiliza los métodos HTTP: GET, POST, PUT y DELETE.
- Es posible llamar servicios basados en OData utilizando una URI y luego interpretar el código XML o JSON aunque existen herramientas y librerías de alto nivel que nos ahorrarán este trabajo.
- Estas librerías tienen la misma función que en SOAP: formar y enviar las peticiones y manejar las respuestas.
- También hay librerías para el lado del servidor.
- http://www.odata.org/libraries/

Peticiones

Peticiones

- Las peticiones se realizan a través de URIs.
- Partes que componen una URI en OData:
  - Protocolo
  - Domain root o raíz del dominio
  - Ruta o path del recurso
  - Parámetros de consulta o Query String Parameters.

```
http://services.odata.org/OData/OData.svc/Customers?$top=5&$orderby=Name
^^^^   ^^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^ ^^^^^^^^^ ^^^^^^^^^^^^^^^^^^^^
prot       domain root     resource path  data coll     parameters
```

Peticiones

- Los parámetros precedidos por un símbolo de Dollar se llaman parámetros de sistema. Estos están incluidos en la especificación de OData.

http://services.odata.org/odata/odata.svc/

Peticiones

Ejemplos:

- http://services.odata.org/OData/OData.svc/Products/?$filter=Price%20ls%205
- http://services.odata.org/OData/OData.svc/Products/?$top=5
- http://services.odata.org/OData/OData.svc/Products/?$filter=Price%20lt%205&$orderby=Price
- http://services.odata.org/OData/OData.svc/Products(4)/Supplier
- http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part2-url-conventions.html

Respuestas

Respuestas

- La respuestas puede contener múltiples formatos.
- El documento raíz describe las colecciones disponibles en el servicio. El formato de este mensaje puede ser XML plano o JSON.
- Los datos pueden ser solicitados en formato ATOM o JSON.

Respuestas

Petición de datos en XML:

- http://services.odata.org/OData/OData.svc/?$format=xml
- http://services.odata.org/OData/OData.svc/?$format=json

Para los datos:

- http://services.odata.org/OData/OData.svc/Products/?$top=3&$format=atom
- http://services.odata.org/OData/OData.svc/Products/?$top=3&$format=json

Respuestas

También es posible especificar el formato del cuerpo del mensaje en un formato concreto utilizando la cabecera *content-type*:

- application/atom+xml
- application/json